[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/22
G06F 9/30

[12] 发明专利申请公开说明书

[21] 申请号 03103038.6

[43] 公开日 2003 年 7 月 23 日

[11] 公开号 CN 1431584A

[22] 申请日 2003.1.28 [21] 申请号 03103038.6

[30] 优先权

[32] 2002. 8.22 [33] US [31] 10/227,008

[71] 申请人 智慧第一公司

地址 美国德克萨斯州

[72] 发明人 G·葛兰·亨利 罗德·E·胡克 泰瑞·派克斯

[74] 专利代理机构 隆天国际知识产权代理有限公司

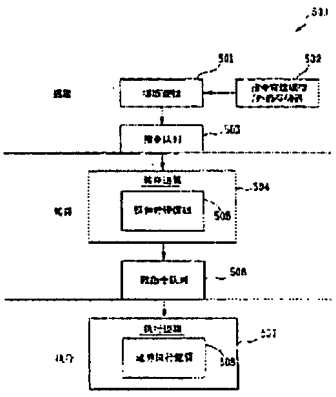
代理人 楼仙英 陈 红

权利要求书 3 页 说明书 18 页 附图 9 页

[54] 发明名称 延仲微处理器数据模式的装置及方法

[57] 摘要

本发明涉及一种装置及方法，用于扩充一微处理器指令集，以提供可由该微处理器指令集的可程序化指令予以指定的延仲大小操作数。该装置包括一转译逻辑与一延仲执行逻辑。该转译逻辑将一延仲指令转译成对应的微指令，由该微处理器执行。该延仲指令具有一延仲前置码与一延仲前置码标记。该延仲前置码指定对应一指定运算的一操作数的延伸操作数大小，其中该延伸操作数大小是无法由一即有指令集进行指定。该延仲前置码标记则指出该延仲前置码，且是原本该即有指令集内另一依据架构所指定的运算码。该延仲执行逻辑耦接至转译逻辑，用以接收该对应的微指令，并使用该操作数以执行该指定运算。



ISSN1008-4274

1．一种用以延伸一微处理器的数据模式的装置，其特征在于，它包括：

一转译逻辑，用以将一延伸指令转译成对应的微指令，由微处理器执行，其中该延伸指令包括：

一延伸前置码，用以指定对应一指定运算的一操作数的延伸操作数大小，其中该延伸操作数大小不能由一既有指令集加以指定；以及

一延伸前置码标记，用以指出该延伸前置码，其中该延伸前置码标记是原本该既有指令集内另一依据架构所指定的运算码；以及

一延伸执行逻辑，耦接至该转译逻辑，用以接收该对应的微指令，并使用该操作数来执行该指定运算。

2．如权利要求1所述的装置，其特征在于所述的延伸指令还包括该既有指令集的指令项目。

3．如权利要求2所述的装置，其特征在于所述的指令项目指定该微处理器所要执行的该运算，且其中对应该运算的该操作数是提取自／储存至一缓存器，其中该缓存器依据数个操作数大小支持操作数的提取／储存。

4．如权利要求1所述的装置，其特征在于所述的延伸前置码指示该微处理器于执行该指定运算时，取代该操作数的一预设的操作数大小。

5．如权利要求1所述的装置，其特征在于所述的延伸前置码包括：

一延伸操作数大小字段，用以指定该操作数的该延伸操作数大小，其中该延伸操作数大小是该些操作数大小之一。

6．如权利要求1所述的装置，其特征在于所述的转译逻辑包括：

一逸出指令检测逻辑，用于检测该延伸前置码标记；

一指令译码逻辑，用以决定该操作数及所要执行的该运算；以及

一延伸译码逻辑，耦接至该逸出指令检测逻辑与该指令译码逻辑，用以决定该延伸操作数大小，并于该对应微指令内指定该延伸操作数大小。

7．一种扩充一既有微处理器指令集以提供延伸数据模式的机制，其特征在于，它包括：

一延伸指令，组态为指定一操作数的延伸操作数大小，该操作数对应一指定运算，而该延伸指令包括该既有微处理器指令集其中一选取的运算码，

其后则接着一 n 位的延伸前置码，该选取的运算码指出该延伸指令，而该 n 位的延伸前置码则指出该延伸操作数大小，其中该延伸操作数大小不能另依该既有微处理器指令集加以指定；以及

一转译器，组态为接收该延伸指令，并产生一微指令序列，以指示一微处理器依照该延伸操作数大小，于该操作数上执行该指定运算。

8．如权利要求 7 所述的机制，其特征在于所述的延伸指令还包括：

其余指令项目，组态为指定该操作数与该指定运算，其中该指定运算的该操作数是提供自／至一延伸操作数缓存器。

9．如权利要求 7 所述的机制，其特征在于所述的 n 位的前置码包括：

一数据模式取代字段，组态为指定该延伸操作数大小予该操作数，其中该延伸操作数大小包括数个操作数大小其中之一。

10．如权利要求 7 所述的机制，其特征在于所述的转译器包括：

一逸出指令检测器，用以检测该延伸指令内的该选取的运算码；

一指令译码器，用以译码该延伸指令的其余部分，以决定该指定运算；以及

一延伸前置码译码器，耦接至该逸出指令检测器及该指令译码器，用以译码该 n 位的延伸前置码，并于该微指令序列内指定该延伸操作数大小。

11．一种为一既有指令集增添延伸数据模式能力的指令集延伸模块，其特征在于，它包括：

一逸出标记，由一转译逻辑接收，并指出一对应指令的附随部分加以指定的一微处理器所要执行一延伸运算，其中该逸出标记为该既有指令集内的一第一运算码；

一延伸操作数大小指定元，耦接至该逸出标记，且为该附随部分其中之一，用以指定对应该延伸运算的数个数据模式其中之一；以及

一延伸执行逻辑，耦接至该转译逻辑，利用所指定的数据模式执行该延伸运算，其中该既有指令集仅提供既有的数据模式，而未能提供所指定的数据模式。

12．如权利要求 11 所述的指令集延伸模块，其特征在于所述的附随部分的其余部分包括一第二运算码与选用的数个地址指定元，用以指定该延伸运算与数个操作数，其中该些操作数是依所指定的数据模式加以执行。

13．如权利要求 11 所述的指令集延伸模块，其特征在于所述的转译逻辑将该逸出标记与该附随部分转译成对应的微指令，该对应的微指令是指示该延伸缓存器逻辑依据所指定的数据模式，于该延伸运算执行时，存取一延伸缓存器，以提取／储存一延伸操作数。

14．如权利要求 11 所述的指令集延伸模块，其特征在于所述的转译逻辑包括：

一逸出标记检测逻辑，用以检测该逸出标记，并指示该附随部分的转译动作需依据延伸转译常规；以及

一译码逻辑，耦接至该逸出标记检测逻辑，用以依据该既有指令集的常规，执行指令的转译动作，并依据该延伸转译常规执行该对应指令的转译，以依据所指定的数据模式，致能该延伸运算的执行。

15．一种扩充一既有指令集架构的方法，可在一微处理器内程序化地指定一延伸数据模式，该方法包括：

提供一延伸指令，该延伸指令包括一延伸标记及一延伸前置码，其中该延伸标记是该既有指令集架构其中一第一运算码项目；

通过该延伸前置码与该延伸指令的其余部分指定该延伸数据模式与一指定运算，其中该既有指令集架构仅提供指定既有数据模式而非该延伸数据模式的指令；以及

依据该延伸数据模式执行该指定运算。

16．如权利要求 15 所述的方法，其特征在于所述的指定延伸数据模式的动作包括：首先指定该指定运算，其中该首先指定的动作包括使用该既有指令集架构中一第二运算码项目。

17．如权利要求 15 所述的方法，其特征在于，还包括：

将该延伸指令转译成微指令，该微指令是指示一延伸执行逻辑依据该延伸数据模式执行该延伸运算。

18．如权利要求 17 所述的方法，其特征在于所述的转译延伸指令的动作包括：

于一转译逻辑内，检测该延伸标记；以及

依照延伸转译规则译码该延伸前置码与该延伸指令的其余部分，以取代该延伸运算的一预设数据模式。

延伸微处理器数据模式的装置及方法

与相关申请案的对照

（0001）本申请主张以下美国申请案的相关权益：案号 10 / 227008，申请日为 2002 年 8 月 22 日。

（0002）本申请与下列同在申请中的美国专利申请案有关，都具有相同的申请人与发明人。

台湾申请案号	申请日	DOCKET NUMBER	专利名称
91116957	7 / 30 / 02	CNTR：2176	延伸微处理器指令集的装置及方法
91116958	7 / 30 / 02	CNTR：2186	执行条件指令的装置及方法
91124008	10 / 18 / 02	CNTR：2187	选择性控制存储器属性的装置及方法
91116956	7 / 30 / 02	CNTR：2188	选择性地控制条件码回写的装置及方法
91116959	7 / 30 / 02	CNTR：2189	增加微处理器的缓存器数量的机制
91124006	10 / 18 / 02	CNTR：2191	延伸微处理器地址模式的装置及方法
		CNTR：2192	储存检查的禁止
		CNTR：2193	选择性中断的禁止
91124007	10 / 18 / 02	CNTR：2195	非暂存存储器参照控制机制
91116672	7 / 26 / 02	CNTR：2198	选择性地控制结果回写的装置及方法

技术领域

（0003）本发明是有关微电子的领域，尤指一种能将延伸地址模式控制纳入一既有的微处理器指令集架构的技术。

背景技术

（0004）自 1970 年代初面世以来，微处理器的使用即呈指数般成长。从最早应用于科学与技术的领域，到如今已从那些特殊领域引进商业的消费者领域，如桌上型与膝上型（laptop）计算机、视频游戏控制器以及许多其它常见的家用与商用装置等产品。

（0005）随着使用上的爆炸性成长，在技术上也历经一相对应的提升，其特征在于对下列项目有着日益升高的要求：更快的速度、更强的寻址能力、更快的存储器存取、更大的操作数，更多种一般用途类型的运算（如浮点运算、单一指令多重数据（SIMD）、条件移动等）以及附加的特殊用途运算（如数字信号处理功能及其它多媒体运算）。如此造就了该领域中惊人的技术进展，且都已应用于微处理器的设计，像扩充流水线化（extensive pipelining）、超纯量架构（super-scalar architecture）、快取结构、乱序处理（out-of-order processing）、爆发式存取（burst access）机制、分支预测（branch predication）以及假想执行（speculative execution）。总之，比起 30 年前刚出现时，现在的微处理器呈现出惊人的复杂度，且具备了强大的能力。

（0006）但与许多其它产品不同的是，有另一非常重要的因素已限制了，并持续限制着微处理器架构的演进。现今微处理器会如此复杂，一大部分得归因于这项因素，即旧有软件的兼容性。在市场考量下，所多制造商选择将新的架构特征纳入最新的微处理器设计中，但同时在这些最新的产品中，又保留了所有为确保兼容于较旧的、即所谓“旧有”（legacy）应用程序所必需的能力。

（0007）这种旧有软件兼容性的负担，没有其它地方，会比在 x86-兼容的微处理器的发展史中更加显而易见。大家都知道，现在的 32 / 16 位的虚拟模式（virtual-mode）x86 微处理器，仍可执行 1980 年代所撰写的 8 位真实模式（real-mode）的应用程序。而本领域的技术人员也承认，有不少相关的架构“包袱”堆在 x86 架构中，只为了支持与旧有应用程序及运作模式的兼容性。虽然在过去，研发者可将新开发的架构特征加入既有的指令集架构，但如今使用这

些特征所凭借的工具，即可程序化的指令，却变得相当稀少。更简单他说，在某些重要的指令集中，已没有“多余”的指令，让设计者可藉以将更新的特征纳入一既有的架构中。

（0008）例如，在 x86 指令集架构中，已经没有任何一未定义的一字节大小的运算码状态，是尚未被使用的。在主要的一字节大小的 x86 运算码图中，全部 256 个运算码状态都已被既有的指令占用了。结果是，x86 微处理器的设计者现在必须在提供新特征与保留旧有软件兼容性两者间作抉择。若要提供新的可程序化特征，则必须分派运算码状态给这些特征。若既有的指令集架构没有多余的运算码状态，则某些既存的运算码状态必须重新定义，以提供给新的特征。因此，为了提供新的特征，就得牺牲旧有软件兼容性了。

（0009）一个持续恶化且困扰微处理器设计者的问题，即是操作数的大小。早期的微处理器设计提供了使用 8 位操作数的 8 位运算。随着应用程序使用的计算日渐复杂，操作数的大小与相关的运算也增加至 16 位。现在用于桌上型／膝上型应用程序的微处理器，已能提供 32 位的操作数／运算。微处理器的操作数／运算的大小，通常称为数据模式（data mode）。因此，为了保留与旧有应用程序的兼容性，现代的桌上型／膝上型计算机的微处理器皆能以 32 位、16 位甚至是 8 位的数据模式运作。

（0010）但即使到现在，由于微处理器不能支持延伸数据模式，如 64 位与 128 位的数据模式，仍有些应用程序的领域会遭受不利的影响。不过，为了要在已无剩余运算码值的架构内支持这些延伸数据模式，必须将既有运算码重新定义，如此将会导致无法支持旧有的应用程序。

（0011）因此，我们所需要的是，一种可将延伸数据模式纳入既有微处理器指令集架构的装置及方法，其中该指令集架构是被已定义的运算码完全占用，且纳入该延伸数据模式还能让一符合旧有规格的微处理器保留执行旧有应用程序的能力。

发明内容

（0012）本发明如同前述其它申请案，是针对上述及其它公知技术的问题与缺点加以克服。本发明提供一种更好的技术，用以扩充微处理器的指令集，使其超越现有的能力，提供延长的操作数，可由该微处理器指令集的可程序化

指令在其上运作。在一具体实施例中，提供了一种用以延伸微处理器数据模式的装置。该装置包括一转译逻辑（translation logic）与一延伸执行逻辑（extended execution logic）。该转译逻辑将一延伸指令转译成对应的微指令（micro instruction），由微处理器执行。该延伸指令具一延伸前置码（extended prefix）与一延伸前置码标记（extended prefix tag）。该延伸前置码指定对应一指定运算的一操作数的延伸操作数大小，其中该延伸操作数大小不能由一既有指令集加以指定。该延伸前置码标记则指出该延伸前置码，其中延伸前置码标记是原本该既有指令集内另一依据架构所指定的运算码。该延伸执行逻辑耦接至转译逻辑，用以接收该对应的微指令，并使用该操作数来执行该指定运算。

（0013）本发明的一个目的在于提出一种扩充既有微处理器指令集以提供延伸数据模式的机制。该机制包括一延伸指令与一转译器（translator）。该延伸指令指定一操作数的延伸操作数大小。该操作数是对应一指定运算，而该延伸指令包括该既有微处理器指令集其中一选取的运算码，其后则接着一 n 位的延伸前置码。该选取的运算码指出该延伸指令，而该 n 位的延伸前置码则指出该延伸操作数大小。该延伸操作数大小不能另依该既有微处理器指令集加以指定。该转译器组态为接收该延伸指令，并产生一微指令序列，以指示微处理器依照该延伸操作数大小，于该操作数上执行该指定运算。

（0014）本发明的另一目的在于提出一种为既有指令集增添延伸数据模式能力的指令集延伸模块。该指令集延伸模块具有一逸出标记（escape tag）、一延伸操作数大小指定元（extended operand size specifier）及一延伸执行逻辑。该逸出标记由一转译逻辑接收，并指出一对应指令的附随部分是指定了微处理器所要执行的一延伸运算，其中该逸出标记为该既有指令集内的一第一运算码。该延伸操作数大小指定元耦接至该逸出标记，且为该附随部分其中之一，用以指定对应该延伸运算的数个数据模式其中之一。该延伸执行逻辑耦接至该转译逻辑，利用所指定的数据模式执行该延伸运算，其中该既有指令集仅提供既有的数据模式，而未能提供所指定的数据模式。

（0015）本发明的再一目的在于提供一种扩充既有指令集架构的方法，可在微处理器内程序化地指定一延伸数据模式。该方法包括提供一延伸指令，该延伸指令包括一延伸标记及一延伸前置码，其中该延伸标记是该既有指令集架构其中一第一运算码项目；通过该延伸前置码与该延伸指令的其余部分指定该

延伸数据模式与一指定运算，其中该既有指令集架构仅提供指定既有数据模式而非该延伸数据模式的指令；以及依据该延伸数据模式执行该指定运算。

附图说明

（0016）本发明的前述与其它目的、特征及优点，在配合下列说明及所附图标后，将可获得更好的理解：

（0017）图 1 为一相关技术的微处理器指令格式的方块图；

（0018）图 2 为一表格，其描述一指令集架构中的指令，如何对应至图 1 指令格式内一 8 位运算码字节的位逻辑状态；

（0019）图 3 为本发明的延伸指令格式的方块图；

（0020）图 4 为一表格，其显示依据本发明，延伸架构特征如何对应至一 8 位延伸前置码实施例中位的逻辑状态；

（0021）图 5 为解说本发明应用延伸数据模式的一流水线化微处理器的方块图；

（0022）图 6 为本发明用于指定一微处理器中的延伸数据模式的延伸前置码的一具体实施例的方块图；

（0023）图 7 为图 5 微处理器内转译阶段逻辑的具体的方块图；

（0024）图 8 为图 5 的微处理器内延伸执行阶段逻辑的方块图；以及

（0025）图 9 为描述本发明对于指定微处理器中的一延伸数据模式运算的指令，进行转译与执行的方法的运作流程图。

图标说明：

100 指令格式	101 前置码
102 运算码	103 地址指定元
200 8 位运算码图	201 运算码值
202 运算码 F1H	
300 延伸指令格式	301 前置码
302 运算码	303 地址指定元
304 延伸指令标记	305 延伸前置码
400 8 位前置码图	401 架构特征

500 流水线化微处理器　　501 提取逻辑

502 指令高速缓存 / 外部存储器

503 指令队列　　504 转译逻辑

505 延伸转译逻辑　　506 微指令队列

507 执行逻辑　　508 延伸执行逻辑

600 延伸前置码　　601 地址模式字段

602 备用字段

700 转译阶段逻辑　　701 激活状态信号

702 机器特定缓存器　　703 延伸特征字段

704 指令缓冲器　　705 转译逻辑

706 转译控制器　　707 除能信号

708 逸出指令检测器　　709 延伸前置码译码器

710 指令译码器　　711 控制只读存储器

712 微指令缓冲器　　713 运算码延伸项字段

714 微运算码字段　　715 目的字段

716 来源字段　　717 位移字段

800 延伸执行阶段逻辑　　801 微指令缓冲器

802 操作数输入缓冲器　　803 操作数输入缓冲器

804 延伸地址逻辑　　805 偏移缓冲器

806 延伸线性地址产生器　　807 节区选取元缓冲器

808 节区描述元表　　809 节区描述元

810 缓冲器　　811 缓冲器

812 较低线性地址缓冲器　　813 较高线性地址缓冲器

814 操作数字段　　815 操作数延伸项字段

816 延伸缓存器

900～924 对指定微处理器的延伸数据模式运算的指令，进行转译与执行的方法的运作流程

100 指令格式　　101 前置码

102 运算码　　103 地址指定元

200 8 位运算码图　　201 运算码值

202 运算码 F1H

300 延伸指令格式　　301 前置码

302 运算码　　303 地址指定元

304 延伸指令标记　　305 延伸前置码

400 8 位前置码图　　401 架构特征

500 流水线化微处理器　　501 提取逻辑

502 指令高速缓存 / 外部存储器

503 指令队列　　504 转译逻辑

505 延伸转译逻辑　　506 微指令队列

507 执行逻辑　　508 延伸执行逻辑

600 延伸前置码　　601 地址模式字段

602 备用字段

700 转译阶段逻辑　　701 激活状态信号

702 机器特定缓存器　　703 延伸特征字段

704 指令缓冲器　　705 转译逻辑

706 转译控制器　　707 除能信号

708 逸出指令检测器　　709 延伸前置码译码器

710 指令译码器　　711 控制只读存储器

712 微指令缓冲器　　713 运算码延伸项字段

714 微运算码字段　　715 目的字段

716 来源字段　　717 位移字段

800 延伸执行阶段逻辑　　801 微指令缓冲器

802 操作数输入缓冲器　　803 操作数输入缓冲器

804 延伸地址逻辑　　805 偏移缓冲器

806 延伸线性地址产生器　　807 节区选取元缓冲器

808 节区描述元表　　809 节区描述元

810 缓冲器　　811 缓冲器

812 较低线性地址缓冲器　　813 较高线性地址缓冲器

814 操作数字段　　815 操作数延伸项字段

816 延伸缓存器

900~924 对指定微处理器的延伸数据模式运算的指令，进行转译与执行的方法的运作流程

具体实施方式

（0026）以下的说明，是在一特定实施例及其必要条件的脉络下而提供，可使一般本领域技术人员能够利用本发明。然而，各种对该较佳实施例所作的修改，对本领域技术人员而言乃是显而易见，并且，在此所讨论的一般原理，亦可应用至其它实施例。因此，本发明并不限于此处所展示与叙述的特定实施例，而是具有与此处所公开的原理与新颖特征相符的最大范围。

（0027）前文已针对今日的微处理器内，如何扩充其架构特征，以超越相关指令集能力的技术，作了背景的讨论。有鉴于此，在图 1 与图 2，将讨论一相关技术的例子。此处的讨论强调了微处理器设计者所一直面对的两难，即一方面，他们想将最新开发的架构特征纳入微处理器的设计中，但另一方面，他们又要保留执行旧有应用程序的能力。在图 1 至 2 的例子中，一完全占用的运算码图，已把增加新运算码至该范例架构的可能性排除，因而迫使设计者要不就选择将新特征纳入，而牺牲某种程度的旧有软件兼容性，要不就将架构上的最新进展一并放弃，以使维持微处理器与旧有应用程序的兼容性。在相关技术的讨论后，于图 3 至 9，将提供对本发明的讨论。藉由利用一既有但未使用的运算码作为一延伸指令的前置码标记，本发明可让微处理器设计者克服已完全使用的指令集架构的限制，除了提供程序员使用比现有还长的操作数执行运算的能力，同时也能保留执行旧有应用程序所需的所有特征。

（0028）请参阅图 1，其是一相关技术的微处理器指令格式 100 的方块图。该相关技术的指令 100 具有数量可变的数据项 101—103，每一项目皆设定成一特定值，合在一起便组成微处理器的一特定指令 100。该特定指令 100 指示微处理器执行一特定运算，例如将两操作数相加，或者是将一操作数从存储器搬移至一内部缓存器，或从该内部缓存器搬移至存储器。一般而言，指令 100 内的运算码项目 102 指定了所要执行的特定运算，而选用（optional）的地址指定元项目 103 位于运算码 102 之后，以指定关于该特定运算的附加信息，像是如何执行该运算，操作数位于何处等等。指令格式 100 并允许程序员在一运算码 102 前加上前置码项目 101。在运算码 102 所指定的特定运算执行时，前

置码 101 用以指示是否使用特定的架构特征。一般来说，这些架构特征能应用于指令集中任何运算码 102 所指定运算的大部分。例如，现今前置码 101 存在于一些能使用不同大小的操作数（如 8 位、16 位、32 位）执行运算的微处理器中。而当许多此类处理器被程序化为一预设的操作数大小时（比如 32 位），在其个别指令集中所提供的前置码 101，仍能使程序员依据各个指令，选择性地取代（override）该预设的操作数大小（如为了执行 16 位的运算）。可选择的操作数大小仅是架构特征的一例，在许多现代的微处理器中，这些架构特征能应用于众多可由运算码 102 加以指定的运算（如加、减、乘、布尔逻辑等）。

（0029）图 1 所示的指令格式 100，有一为业界所熟知的范例，此即 x86 指令格式 100，其为所有现代的 x86-兼容微处理器所采用。更具体他说，x86 指令格式 100（也称为 x86 指令集架构 100）使用了 8 位前置码 101、8 位运算码 102 以及 8 位地址指定元 103。x86 架构 100 亦具有数个前置码 101，其中两个取代了 x86 微处理器所预设的地址 / 数据大小（即运算码状态 66H 与 67H），另一个则指示微处理器依据不同的转译规则来解译其后的运算码字节 102（即前置码值 0FH，其使得转译动作是依据所谓的二字节运算码规则来进行），其它的前置码 101 则使特殊运算重复执行，直至重复条件满足为止（即 REP 运算码：F0H、F2H 及 F3H）。

（0030）现请参阅图 2，其显示一表格 200，用以描述一指令集架构的指令 201 如何对应至图 1 指令格式内一 8 位运算码字节 102 的位值。表格 200 呈现了一 8 位运算码图 200 的范例，其将一 8 位运算码项目 102 所具有的最多 256 个值，关联到对应的微处理器运算码指令 201。表格 200 将运算码项目 102 的一特定值，譬如 02H，映像至一对应的运算码指令 201（即指令 I02 201）。在 x86 运算码图的例子中，为此领域中人所熟知的是，运算码值 14H 是映像至 x86 的进位累加（Add with Carry，ADC）指令，此指令将一 8 位的直接（immediate）操作数加至架构缓存器 AL 的内含值。本领域的技术人员也将发觉，上文提及的 x86 前置码 101（亦即 66H、67H、0FH、F0H、F2H 及 F3H）是实际的运算码值 201，其在不同脉络下，指定要将特定的架构延伸项应用于随后的运算码项目 102 所指定的运算。例如，在运算码 14H（正常情况下，是前述的 ADC 运算码）前加上前置码 0FH，会使得 x86 处理器执行一“解压缩与插入低压缩的单精度浮点值”（Unpack and Interleave Low Packed

Single-Precision Floating-Point Values）运算，而非原本的 ADC 运算。诸如此 x86 例子所述的特征，在现代的微处理器中是部分地致能，此因微处理器内的指令转译解码逻辑是依序解译一指令 100 的项目 101－103。所以在过去，于指令集架构中使用特定运算码值作为前置码 101，可允许微处理器设计者将不少先进的架构特征纳入兼容旧有软件的微处理器的设计中，而不会对未使用那些特定运算码状态的旧有程序，带来执行上的负面冲击。例如，一未曾使用 x86 运算码 0FH 的旧有程序，仍可在今日的 x86 微处理器上执行。而一较新的应用程序，借着运用 x86 运算码 0FH 作为前置码 101，就能使用许多新进纳入的 x86 架构特征，如单一指令多重数据（SIMD）运算，条件移动运算等等。

（0031）尽管过去已藉由指定可用／多余的运算码值 201 作为前置码 10i（也称为架构特征标记／指针 101 或逸出指令 10i），来提供架构特征，但许多指令集架构 100 在提供功能上的强化时，仍会因为一非常直接的理由，而碰到阻碍：所有可用／多余的运算码值已被用完，也就是，运算码图 200 中的全部运算码值已被架构化地指定。当所有可用的值被分派为运算码项目 102 或前置码项目 101 时，就没有剩余的运算码值可作为纳入新特征之用。这个严重的问题存在于现在的许多微处理器架构中，因而迫使设计者得在增添架构特征与保留旧有程序的兼容性两者间作抉择。

（0032）值得注意的是，图 2 所示的指令 201 是以一般性的方式表示（亦即 124、186），而非具体指涉实际的运算（如进位累加、减、异或）。这是因为，在一些不同的微处理器架构中，完全占用的运算码图 200 在架构上，已将纳入较新进展的可能性排除。虽然图 2 例子所提到的，是 8 位的运算码项目 102，本领域的技术人员仍将发觉，运算码 102 的特定大小，除了作为一特殊情况来讨论完全占用的运算码结构 200 所造成的问题外，其它方面与问题本身并不相干。因此，一完全占用的 6 位运算码图将有 64 个可架构化地指定的运算码／前置码 201，并将无法提供可用／多余的运算码值作为扩充之用。

（0033）另一种替代做法，则并非将原有指令集完全废弃，以一新的格式 100 与运算码图 200 取代，而是只针对一部份既有的运算码 201，以新的指令意含取代，如图 2 的运算码 40H 至 4FH。以这种混合的技术，微处理器就可以单独地以下列两种模式的一运作：其中旧有模式利用运算码 40H－4FH，是依旧有规则来解译，或者以另一种改良模式（enhanced mode）运作，此时运

算码 40H—4FH 则依加强的架构规则来解译。此项技术确能允许设计者将新特征纳入设计，然而，当符合旧有规格的微处理器于加强模式运作时，缺点仍旧存在，因为微处理器不能执行任何使用运算码 40H—4FH 的应用程序。因此，站在保留旧有软件兼容性的立场，兼容旧有软件 / 加强模式的技术，还是无法接受的。

（0034）然而，对于运算码空间已完全占用的指令集 200，且该空间涵盖所有于符合旧有规格的微处理器上执行的应用程序的情形，本案发明人已注意到其中运算码 201 的使用状况，且他们亦观察出，虽然有些指令 202 是架构化地指定，但未用于能被微处理器执行的应用程序中。图 2 所述的指令 IF1 202 即为此现象的一例。事实上，相同的运算码值 202（亦即 F1H）是映像至未用于 x86 指令集架构的一有效指令 202。虽然该未使用的 x86 指令 202 是有效的 x86 指令 202，其指示要在 x86 微处理器上执行一架构化地指定的运算，但它却未使用于任何能在现代 x86 微处理器上执行的应用程序。这个特殊的 x86 指令 202 被称为电路内模拟断点（In Circuit Emulation Breakpoint）（亦即 ICE BKPT，运算码值为 F1H），之前都是专门使用于一种现在已不存在的微处理器模拟设备中。ICE BKPT 202 从未用于电路内模拟器之外的应用程序中，并且先前使用 ICE BKPT 202 的电路内模拟设备已不复存在。因此，在 x86 的情形下，本案发明人已在一完全占用的指令集架构 200 内发现一样工具，借着利用一有效但未使用的运算码 202，以允许在微处理器的设计中纳入先进的架构特征，而不需牺牲旧有软件的兼容性。在一完全占用的指令集架构 200 中，本发明利用一架构化地指定但未使用的运算码 202，作为一指针标记，以指出其后的一 n 位前置码，因此允许微处理器设计者可将最多 2^n 个最新发展的架构特征，纳入微处理器的设计中，同时保留与所有旧有软件完全的兼容性。

（0035）本发明藉提供一 n 位的延伸操作数大小指定元前置码，以使用前置码标记 / 延伸前置码的概念，因而可允许程序员在一微处理器中，依据每个指令指定一延伸数据模式予一对应的运算。该延伸数据模式是用以取代该微处理器的既有指令集架构所支持的既有数据模式。本发明现将参照图 3 至 9 进行讨论。

（0036）现请参阅图 3，其为本发明的延伸指令格式 300 的方块图。与图 1 所讨论的格式 100 非常近似，该延伸指令格式 300 具有数量可变的指令项目

301—305，每一项目设定为一特定值，集合起来便组成微处理器的一特定指令300。该特定指令300指示微处理器执行一特定运算，像是将两操作数相加，或是将一操作数从存储器搬移至微处理器的缓存器内。一般而言，指令300的运算码项目302指定了所要执行的特定运算，而选用的地址指定元项目303则位于运算码302后，以指定该特定运算的相关附加信息，像是如何执行该运算，操作数位于何处等等。指令格式300亦允许程序员在一运算码。302前加上前置码项目301。在运算码302所指定的特定运算执行时，前置码项目301是用来指示是否要使用既有的架构特征。

（0037）然而，本发明的延伸指令300是前述图1指令格式100的一超集（superset），其具有两个附加项目304与305，可被选择性作为指令延伸项，并置于一格式化延伸指令300中所有其余项目301—303之前。这两个附加项目304与305可让程序员能在一符合旧有规格的微处理器内，指定一延伸数据模式，以依据该延伸数据模式执行一运算，其中该延伸数据模式是无法另由该符合旧有规格微处理器的既有指令集来加以程序化。这两个附加项目304与305可将较大的操作数／运算纳入一具有已完全占用的指令集架构的微处理器设计中，选用项目304与305是一延伸指令标记304与一延伸操作数大小指定元前置码305。该延伸指令标记304是一微处理器指令集内另一依据架构所指定的运算码。在一x86的实施例中，该延伸指令标记304，或称逸出标记304，是用运算码状态F1H，其为早先使用的ICE BKPT指令。逸出标记304向微处理器逻辑指出，该延伸前置码305，或称延伸特征指定元305，是跟随在后，其中该延伸前置码305指定了对应于一指定运算的一操作数／运算大小或数据模式。在一具体实施例中，逸出标记304指出，一对应延伸指令300的附随部分301—303及305指定了微处理器所要执行的一延伸运算。延伸操作数大小指定元305，或称延伸前置码305，指定了对应于一相关运算的数个操作数大小其中之一。微处理器内的延伸执行逻辑于执行该延伸运算时，存取延伸大小的缓存器中的操作数，并使用与该指定的操作数大小或数据模式相一致的处理规则，来处理所存取的操作数。

（0038）此处将本发明的延伸数据模式的技术作个概述。一延伸指令是组态为于一既有微处理器指令集中指定一延伸数据模式，其中该延伸数据模式无法另依该既有微处理器指令集来加以指定。该延伸指令包括该既有指令集的运

算码 / 指令 304 其中之一以及一 n 位的延伸特征前置码 305。所选取的运算码对旨令作为一指针 304，以指出指令 300 是一延伸特征指令 300（亦即，其指定了微处理器架构的延伸项），该 n 位的特征前置码 305 则指出该延伸数据模式。在一具体实施例中，延伸前置码 305 具有八位，最多可指定 256 种不同的数据模式。n 位前置码的实施例，则最多可指定 2^n 种不同的数据模式。在另一实施例中，提供 64 位的数据模式，以取代符合旧有规格微处理器中预设的数据模式（如 32 位或 16 位）。因此，在对应的运算执行时，执行逻辑即于 64 位的操作数上执行 64 位的运算（如加、减、逻辑运算等）。在另一实施例中，则更允许程序员指定 64 位或 128 位的数据模式。

（0039）现请参阅图 4，一表格 400 显示依据本发明，缓存器延伸项如何映像至一 8 位延伸前置码实施例的位逻辑状态。类似于图 2 所讨论的运算码图 200，图 4 的表格 400 呈现一 8 位的延伸数据模式前置码图 400 的范例，其将一 8 位延伸前置码项目 305 的最多 256 个值，关联到一符合旧有规格的微处理器的对应延伸数据模式 401（如 E34、E40 等）。在一 x86 的具体实施例中，本发明的 8 位延伸特征前置码 305 是提供给数据模式 401（亦即 E00-RFF）之用，该些数据模式 401 乃现行 x86 指令集架构所未能提供的。

（0040）图 4 所示的延伸特征 401 是以一般性的方式表示，而非具体指涉实际的特征，此因本发明的技术可应用于各种不同的架构延伸项 401 与特定的指令集架构。本领域的技术人员将发觉，许多不同的架构特征 401，其中一些已于上文提及，可依此处所述的逸出标记 304 / 延伸前置码 305 技术将其纳入一既有的指令集。图 4 的 8 位前置码实施例提供了最多 256 个不同的特征 401，而一 n 位前置码实施例则具有最多 2^n 个不同特征 401 的程序化选择。

（0041）现请参阅图 5，其为解说本发明用以执行延伸数据模式运算的流水线化微处理器 500 的方块图。微处理器 500 具有三个明显的阶段类型：提取。转译及执行。提取阶段具有提取逻辑 501，可从指令高速缓存 502 或外部存储器 502 提取指令。所提取的指令经由指令队列 503 送至转译阶段。转译阶段具有转译逻辑 504，耦接至一微指令队列 506。转译逻辑 504 包括延伸转译逻辑 505。执行阶段则有执行逻辑 507，其内具有延伸执行逻辑 508。

（0042）依据本发明，于运作时，提取逻辑 501 从指令高速缓存 / 外部存储器 502 提取格式化指令，并将这些指令依其执行顺序放入指令队列 503 中。

接着从指令队列 503 提取这些指令，送至转译逻辑 504。转译逻辑 504 将每一送入的指令转译 / 译码为一对应的微指令序列，以指示微处理器 500 去执行这些指令所指定的运算。依本发明，延伸转译逻辑 505 检测那些具有延伸前置码标记的指令，以进行对应延伸数据模式指定元前置码的转译 / 译码。在一 x86 的实施例中，延伸转译逻辑 505 组态为检测其值为 F1H 的延伸前置码标记，其是 x86 的 ICE BKPT 运算码。延伸微指令字段则提供于微指令队列 506 中，以允许在微处理器 500 内指定延伸数据模式。

（0043）微指令从微指令队列 506 被送至执行逻辑 507，其中延伸执行逻辑 508 组态为依照该延伸微指令字段所指定，存取内部的微处理器缓存器。数个被指定要用于执行一指定运算的来源操作数，则从来源操作数延伸缓存器中提取。延伸执行逻辑 508 执行微指令所指定的运算，并产生对应的结果。随着结果的产生，延伸执行逻辑 508 将该对应结果回写至该。延伸微指令字段所指定的目的操作数延伸缓存器。

（0044）本领域的技术人员将发现，图 5 所示的微处理器 500 是现代的流水线化微处理器 50 经过简化的结果。事实上，现代的流水线化微处理器 500 最多可包括有 20 至 30 个不同的流水线阶段。然而，这些阶段可概括地归类为方块图所示的三个阶段，因此，图 5 的方块图 500 可用以点明前述本发明实施例所需的必要组件。为了简明起见，微处理器 500 中无关的组件并未显示出来。

（0045）现请参阅图 6，其为本发明用于指定一微处理器延伸操作数 / 运算的延伸前置码 600 的一具体实施例的方块图。延伸操作数 / 运算指定元前置码 600 具 8 位大小。在一具体实施例中，8 位前置码 600 的值指定一对应运算的一延伸数据模式，其中该对应运算是由本发明的延伸指令的其余部分所指定，如此处所述。在一 x86 的实施例中，该延伸数据模式（如 64 位的操作数 / 运算）被指定，以取代一预设的数据模式（如 32 位的操作数 / 运算）。

（0046）在图 6 的本发明延伸前置码 600 的实施范例中，整个前置码 600 是用于指定一延伸数据模式。然而，本领域技术人员将察觉，指定数个延伸数据模式其中之一所需的位数，是依该些延伸数据模式的数量而定。因此，一个能够指定 64 位或 128 位数据模式的实施例，仅需前置码 600 的一个位就足以区分该两种模式。所以，前置码 600 的其余位便可用于指定一既有指令集架构所无法提供的其它延伸特征。

（0047）现请参阅图 7，其为图 5 的微处理器内转译阶段逻辑 700 的具体的方块图。转译阶段逻辑 700 具有一指令缓冲器 704，依本发明，其提供延伸指令至转译逻辑 705。转译逻辑 705 是耦接至一具有一延伸特征字段 703 的机器特定缓存器（machine specific register）702。转译逻辑 705 具一转译控制器 706，其提供一除能信号 707 至一逸出指令检测器 708 及一延伸译码器 709。逸出指令检测器 708 耦接至延伸译码器 709 及一指令译码器 710。延伸译码器 709 与指令译码逻辑 710 存取一控制只读存储器（ROM）711，其中储存了对应至某些延伸指令的样板（template）微指令序列。转译逻辑 705 亦包括一微指令缓冲器 712，其具有一运算码延伸项字段 713、一微运算码字段 714、一目的字段 715、一来源字段 716 以及一位移字段 717。

（0048）运作上，在微处理器通电激活期间，机器特定缓存器 702 内的延伸字段 703 的状态是藉由信号激活状态（signal power-up state）701 决定，以指出该特定微处理器是否能转译与执行本发明之用以提供微处理器的延伸数据模式的延伸指令。在一具体实施例中，信号 701 从一特征控制缓存器（图上未显示）导出，该特征控制缓存器则读取一于制造时即已组态的熔丝数组（fuse array）（未显示）。机器特定缓存器 702 将延伸特征字段 703 的状态送至转译控